Exploiting Re-ranking Information in a Case-Based Personal Travel Assistant[1]

Lorcan Coyle and Pádraig Cunningham

Department of Computer Science
Trinity College Dublin
{Lorcan.Coyle, Padraig.Cunningham}@cs.tcd.ie

Abstract. Intelligent software assistants are becoming more common in e-commerce applications. We introduce the Personalised Travel Assistant, an application currently under development that uses case based reasoning to assist the user in arranging flights. It offers a personalised service to its users by automatically learning their travel preferences by exploiting implicit user feedback. When the user selects an offer, the PTA establishes a preference ordering among the whole set of presented offers. This ordering is used to generate rating scores for these offers which are then stored in the user profile as cases. The ordering is also used to refine the user's overall travel preferences by altering their personal similarity measure. The novelty of our approach lies in the application of re-ranking techniques in the automatic rating of travel offers and in the optimisation of similarity measures.

1 Introduction

Most e-commerce stores sell products without the intervention of a human sales assistant. In the absence of human sales assistants there is a need for intelligent software assistants to facilitate the sales process. Most of these assistants help the user to search through a catalogue of products to find an appropriate item. The user makes an initial request for an item and is assisted by the sales assistant until the optimal item is presented to the user. The main goal of these assistants is to minimise the cognitive load on the customers. Assistants that bore or frustrate the customer risk driving them to competing stores, which are never more than one click away. One way to maximise the user's satisfaction is to provide a personalised service. This is achieved if the assistant learns and applies the user's personal preferences in its interactions with the user. These preferences can be discovered by asking the user to explicitly rate individual items or to explain their motivations behind making a particular selection, however we feel that this goes against the goal of reducing cognitive load. Hence our goal is to learn these preferences implicitly and apply them surreptitiously.

There is much work being done on learning a user's ranking criteria by observing their selections from a list of presented items [20,6,12]. Stahl uses a technique called

[1] The support of the Informatics Research Initiative of Enterprise Ireland is gratefully acknowledged.

case order feedback [20] and Branting uses a similar approach called *LCW* (Learning Customer Weights) [6]. These approaches compare the user's preferential ordering of a set of presented items with the recommender system's predicted ordering. They attempt to minimise the error between the predicted and observed ordering by altering the recommender system's similarity measures. One major advantage of using user feedback is that it incorporates utility [3,5] into the similarity measure.

Our work centres on the development of a Personal Travel Assistant (PTA). This is an intelligent sales assistant that helps the user in making flight arrangements. This application is based on a scenario developed by the Foundation for Intelligent Physical Agents (FIPA) in the travel domain [10]. Our application takes a travel request from the user and forwards it to a number of real-world travel agents. These return a number of suitable offers and the PTA recommends the best of these to the user and assists in the selection of an optimal offer for purchase. This paper documents our approach towards the elicitation of individual user preferences to better complete this offer recommendation task.

There are a number of domain specific constraints that should be noted before we explain our implementation. The first is in dealing with the real-world travel providers; these take their requests in a pre-defined manner, there are a number of parameters that must be filled with each request. This restricts the type of requests the PTA can make on behalf of the user. The second is that there is a turnaround time cost associated with making a request to a real-world broker; we have observed these delays to be as much as 30-40 seconds. These factors rule-out the use of a pure iterative conversational CBR approach [15] to initial query refinement since the request is not negotiable, and the user is unlikely to have the patience to continually refine a search until the optimal offer is found. To overcome this, we make the assumption that all users of the system are willing and able to make a broad initial travel request with the standard parameters required by online travel agents (i.e. origin, destination, departure date and number of tickets required).

Another major issue with our implementation is that we are targeting our application at mobile web users, e.g. WAP enabled phones and mobile-PDA users. These devices are constrained in the amount of screen-size they have. This means that we have to be even more sensitive to the information overload problem and minimise the click-distance of the user to their optimal offer. This reinforces our desire to reduce cognitive load on the user. Ideally we would like to present the user with a small group of offers with maximal utility. They could then search through the offers by using a *more-like-this* (MLT) [14,7,18] type of approach using this initial set of offers as a jumping off point.

Our approach is similar to Ricci et al.'s *ITR* (Intelligent Travel Recommender) system [17]. ITR allows the retrieval of a bag of travel products in response to a travel query, while our PTA only deals with a single product type - flights. The similarity based ranking process we use when presenting flights to the user is similar to the approach used in ITR. The differences in approach are as follows: ITR allows the user to explicitly state their preferences and rates travel solutions based on final selections; our PTA system learns user preferences implicitly and rates flights automatically (including those that were not selected by the user).

Section 2 describes our application and how CBR is used in the recommendation of travel offers to the user. It also mentions the possibility of using a collaborative CBR approach to improve problem coverage. Sections 3 and 4 describe the automatic learning techniques in this application. In Section 3 we describe how implicit user feedback is used to rank travel offers. In Section 4 we describe how overall user preferences are learned by the PTA based on these rankings. In Section 5 we describe the current implementation status and outline future research goals. Finally, in Section 6 we conclude the paper.

2 Description of the Application

This section describes the architecture of our PTA application and is described in more detail in [8]. Our PTA assists the user in the planning and selection of flights and acts as the user's proxy when dealing with real world travel providers. It takes travel requests from users and negotiates with real online broker agents for suitable travel offers using a technique known as web-scraping [2]. This involves faking an interaction between a browser and an online flights provider. The PTA fills in fields of a website travel request form with values from the user request and sends it to the web server as a regular HTTP request. When the response is received, the HTML is parsed to get the travel offers.

In order to become efficient at its job, the PTA should be able to learn the user's travel preferences and apply this knowledge when brokering offers. Since the PTA acts on behalf of many users it must build a user model or profile for every user. This model is made up of two types of information, a representation of the user's preferences with respect to previously viewed travel offers, and a representation of the user's overall travel preferences. With every user interaction this model is refined based on the user's selection of flights.

The user interacts with the PTA via a simple web interface. A user interaction consists of a dialog between the user and the PTA. This dialog begins with the user making an initial travel request. This returns a large number of matching travel offers. These offers are ranked and filtered by the PTA according to the user's preferences. A reduced set of offers is presented to the user who selects their preferred offer. The PTA then connects the user to the broker that made the original offer and bows out of the transaction. If no suitable offer is contained in this set of offers, the cycle is repeated until the user is satisfied.

The CBR Solution to Recommending Offers

Personalisation on the PTA is achieved using CBR. The PTA builds a model of each user of the system. This model consists of two parts: a set of cases representing previously viewed travel offers and users' responses to them; and a description of the user's overall travel preferences in the form of similarity measure information. The personalisation process can be broken down into three parts: the ranking of received offers with respect to users' preferences; the storage of presented offers and their

ratings (as new cases); and the refinement of users' profiles with respect to their feedback on presented offers.

When a user makes a new travel request, the PTA generates a case base of similar offers to the current query and sends the request to the broker agents. The brokers return a number of offers that satisfy this request. The PTA passes each of these through a recommender module. Each offer is treated as a target case and passed into the CBR module. A similarity mechanism returns the most similar case(s) from the case base of previously rated offers. The new offer is assigned a prediction score based on the average ratings of these retrieved cases. When all new offers have been scored in this way, the ones with the highest prediction scores are presented to the user as recommended travel offers.

The next stage of the interaction involves the user searching through the available offers and purchasing one or making a modified request. Assuming that the user selects a flight and proceeds to book it, we can draw some inferences about their preferences with regard to the currently presented offers. This inference is explained in more detail in Section 3. Finally, we attempt to draw inferences from this selection into the user's overall travel preferences. This is described in more detail in Section 4.

Collaborative CBR

CBR is only useful in solving a problem if the case base offers good coverage for that problem. If a user encounters a problem outside their previous experience we borrow cases from her neighbours and use a collaborative CBR [13] approach. This should increase the probability of finding a good solution. We use this collaborative approach only when the user's own case base provides inadequate coverage. We calculate problem coverage by looking at the degree of similarity between the current problem case and the most similar cases from the user's own case base. A high level of similarity would indicate good coverage for the current problem. By sharing cases between users we remove the bootstrap problem, whereby the PTA cannot recommend offers for new users because their case bases are empty.

3 Learning User Preferences on Presented Items

In Section 2 we discussed how we intend to recommend new offers to users by using a CBR approach. This approach depends on the user rating all viewed offers. However, since one of our goals in this application is the reduction of the user's cognitive load we want to measure the user's ratings implicitly. We do this by observing which offer the user selects to buy. From this selection we establish a preference ordering between the set of viewed offers. We use this ordering to predict ratings for each offer. This type of feedback works at a local, case level and tells us little about the users overall travel preferences. There follows a description of our approach to learning local user preferences by observing user feedback to presented sets of cases. We then describe how we use this ordering to rate presented offers.

We define the preference ordering as follows: If the user prefers offer i (O_i) to offer j (O_j), we define the following order:

$$O_i > O_j \quad (1)$$

Consider a set of offers that were viewed by the user $S = \{O_1, ..., O_n\}$. If the user selected (i.e. purchased) the i^{th} offer (O_i) we can determine the following preference order:

$$O_i > \{O_1, ..., O_{i-1}, O_{i+1}, ..., O_n\} \quad (2)$$

However, this tells us nothing about the relationship between the offers that weren't selected by the user. By looking at the similarity of each of the offers to the selected offer we can estimate a preferential relationship between them. If we reorder the set of offers according to each offer's similarity (*Sim*) to O_i, we get the classical similarity based ordering:

$$O_1 > O_2 \text{ if } Sim(O_1, O_i) > Sim(O_2, O_i) \quad (3)$$

This estimation is a good indication of the user's real preference ordering because our similarity measure incorporates offer utility (see Section 4). Confirmation of this will require evaluation with real user data. We use this preference ordering to generate user ratings to store in the user's case base of previous offers. Our system uses a rating scale of zero to ten. The offer that was bought by the user (O_i), is given a rating score of ten, and the offer with the lowest similarity to O_i, Sim_{lowest}, is given a score of zero. The other offers are mapped onto the scale using the following formula:

$$Rating_i = 10 \times \frac{Sim_i - Sim_{lowest}}{1 - Sim_{lowest}} \quad (4)$$

Since the PTA broadly searches for travel offers on behalf of the user, we can expect to get a large number of offers. Even after passing these offers through our recommendation system we can expect to have a large set of recommended offers. In order to maximise coverage of the set of all offers we propose to impose a diversity constraint [5] on this initial set (we only intend to impose this constraint on the initial set of offers). We propose to send all offers to the user and allow her to search for an optimal offer on the client side. This *fat-client* approach is similar to those used by [11,22,23]. By recording the iterative selections made by the user in their search as well as the selection of an optimal offer, we can rate all offers that have been viewed. We intend to supplement this MLT strategy with a *less-like-this* (LLT) strategy [14], e.g. a user could ask for more offers that are more like offer *j* (O_j) and less like offer *k* (O_k). We use the same ratings function as with the pure MLT strategy but we change the underlying similarity functions. We create a new similarity function ($Sim_{i\ not\ j}$), which is equal to offer *i*'s similarity to offer *k* (the least preferred offer) minus its similarity to offer *j* (the preferred case), i.e.:

$$Sim_{i\ not\ j}(O_i, O_j, O_k) = Sim(O_i, O_j) - Sim(O_i, O_k) \quad (5)$$

We need to consider the possibility that none of the brokers returned an optimal offer. In this case our PTA will have to broaden the original travel request. In doing this we force the user to complete a new travel request and to endure the delay of

waiting for the brokers to fetch a new batch of offers. Clearly this inconveniences the user but we see it as an inherent problem with the application.

4 Automated Learning of Overall User Preferences

Section 3 described our approach to learning the preference relationship between the travel offers presented to the user. It showed how we use this relationship to rate offers implicitly. However, this approach cannot tell us anything about a user's underlying preferences with respect to why one travel offer is preferable to another. This section describes how we use local user preference relationships (i.e. between individual sets of offers) to learn overall user preferences at the feature level, i.e. the attributes of a travel offer that are important to a user in selecting an optimal offer.

Before we talk about how our system learns overall user preferences, we should give some information as to the structure of a travel offer case. Our travel offer case contains *n* features, the most important of which are the origin, destination, travel time, distance, number of hops and price. In order to calculate similarity between a query and a case (or between two cases) we use a similarity measure that combines the *n* local similarity measures (at the feature level) and feature weights (that represent the relative importance of features):

$$Sim(O_j, O_k) = \sum_{i=1}^{n} w_i \times Sim(f_i^j, f_i^k) \quad (6)$$

We mentioned in the introduction that our user profile contains the user's overall travel preferences. This information is a representation of that user's similarity measure. This information is stored in CBML format [9], an XML-based case representation language. CBML allows for the creation of a user-profile document that specifies certain similarity measure parameters for each user, i.e. feature weights. Fig. 1 shows an example CBML user profile document.

To optimise this function we can learn and alter the feature weights (w_i). There has been much work done [19,20,6,12] in this area that is applicable to our application. Branting describes a procedure for learning users' preferred feature weights with respect to item recommendation by observing their selections from return sets. This approach can be summarised as follows: whenever a user makes a request for an item, the system recommends a set of items. These items are ordered and one or more of them is considered optimal by the system. If the user selects a non-optimal item, the system's prediction was flawed. To correct this, the system alters its feature weights such that the system's recommendation of an optimal item matches the user's selection. Stahl et al. [21] takes this one step further by attempting to learn both the feature weighting and local similarity measures.

One of the main issues with learning similarity measures is that it is very CPU intensive. Stahl [15] and Branting [6] use error functions to describe the deviation between the predicted case order and the observed case order. This error is minimised by altering the similarity measure parameters. This can be done iteratively, e.g. [6], or

by using evolutionary algorithms, e.g. [21,12]. We propose to use an iterative approach to learning an optimal set of feature weightings from each feedback cycle. The user model will then be modified to reflect these observed preferences.

```
<domain name="pta">
  <profile username="default">
      <relevance>
          <feature path="/traveloffer/source" value="2"/>
          <feature path="/traveloffer/price" value="8"/>
          <feature path="/traveloffer/origin" value="5"/>
          <feature path="/traveloffer/destination" value="5"/>
          <feature path="/traveloffer/departuredate" value="8"/>
          <feature path="/traveloffer/distance" value="1"/>
          <feature path="/traveloffer/flighttime" value="1"/>
          <feature path="/traveloffer/hops" value="12"/>
      </relevance>
  </profile>
  <profile username="Coyle" extends="default">
      <relevance>
          <feature path="/traveloffer/source" value="0"/>
          <feature path="/traveloffer/price" value="12"/>
          <feature path="/traveloffer/hops" value="10"/>
      </relevance>
  </profile>
  <profile username="Newbie" extends="default"/>
</domain>
```

Fig. 1. This figure shows an example CBML user profile document. This document describes the feature weights for three users, *default*, *Coyle* and *Newbie*. The default profile specifies feature weights on eight features. Both *Coyle* and *Newbie* extend the default profile by inheriting the feature weights of the default profile. User *Coyle* has overridden three of the feature weights from the default profile, while user *Newbie* inherits the default profile exactly.

Branting's work on learning feature weights in combination with an MLT search strategy yielded results that suggest that there is little to be gained in observing the user's choices beyond the first iteration of the search [6]. Since our PTA uses a similar approach to this, we should investigate his claim. When the PTA presents an initial set of offers to the user they are a diverse set. Assuming that the user asks to see offers that are more like a particular item, our next set of presented offers will be far less diverse. This suggests that we have most to learn from the user's selection from the first set of offers, which would support Branting's results. However, because subsequent sets of offers have lower diversity (since we only impose diversity constraints on the initial set of presented offers) it is possible that we can learn some important preferences with respect to individual features. We propose to do some further investigation into this hypothesis.

5 Current Implementation Status and Future Work

The current status of our implementation of our PTA is that of a skeletal application. Our user-interface is a simple web-interface tailored for the desktop user. Our PTA is

currently capable of accepting an initial travel request from a user and sending it to a single real world travel broker for satisfaction. The recommendation techniques described in Section 2 have been implemented and the PTA can offer a reduced set of the most suitable travel offers to the user based on their case-base of previously viewed offers. We are currently in the process of implementing the feature weight learning algorithm. These components will make up a proof-of-concept prototype application. At this point we intend to develop interfaces into further flight brokers, provide a user friendly front-end, and get real users to interact with the system. Once we have users in the system we will be able to harvest data and perform an evaluation of our techniques. Our evaluation will involve a number of tests:

- User Selections: Over time, the observed user selections should closely match our predictions.
- Click Distance: Low click distances to the optimal offer (the final offer chosen by the user) should be indicative of user satisfaction.
- Stabilisation of overall user preferences: if our user preferences tend to stabilise over time (rather than fluctuate widely with every interaction), we can be confident that they are representative of the user's real preferences

The automated learning processes and CBR approach to recommending offers described in this paper assume that user's preferences are constant and static. Context is not taken into account by the PTA at any point in a user interaction. It may be appropriate to look into incorporating context into the transaction, e.g. the physical location of the user or a measure of their urgency (the amount of time between the request time and the requested departure time). We predict that these features will prove to be important in analysing the utility of travel offers.

We mentioned in Section 2 that a collaborative CBR approach would be used to share travel-offer cases when a user's case coverage is low. The sharing of cases between users has been well documented e.g. [13]. Less work has been done in the area of sharing the more subtle similarity preferences. Minor et al. [16] investigates the sharing of similarity relationships between agents. We intend to investigate their approach and use it to share elements of our user profile documents.

We mentioned in Section 2 that our iterative-search approach could potentially harvest a large number of cases from every user interaction with the PTA. We intend to put some effort into case base maintenance and examine the possibility of only storing a subset of the presented offers from each interaction. This will be especially significant if the cases are extremely similar.

6 Conclusions

We have been working on a Personal Travel Assistant, an intelligent sales assistant that helps the user in making flight arrangements on behalf of its users. The main task of the PTA is the recommendation of travel offers to the user. The PTA uses case based reasoning to complete this task by comparing new offers with similar previously viewed and rated offers. If the most similar previous offer has a high rating the new offer will be recommended to the user by the PTA. This task is carried out ac-

cording to the user's travel preferences. User's travel preferences are made up of two parts: a case base representing previously viewed and rated travel offers; and a profile document describing the user's similarity function (i.e. its feature weights). There are three parts to the recommendation task:

- The learning of a user's preference ordering in a set of presented offers
- The implicit rating of viewed offers based on this ordering
- The optimising of overall user preferences based on preference orderings

Since we aim to reduce cognitive load on the user (and due to screen size constraints) we want to automatically rate offers without explicit user feedback. Section 3 describes our approach to the learning of the preference ordering. It then goes on to describe how this ordering is used to implicitly rate the presented offers. These offers and their ratings are then stored in the user's profile as travel-offer cases.

Section 4 describes our approach to the refinement of overall user preferences by using an approach similar to [6,20]. This user-feedback approach to refinement guarantees the incorporation of utility into the similarity measures. Section 5 briefly describes the current implementation status of our application and goes on to describes some of the work that we intend to do in the coming months.

References

[1] Aha, D. & Breslow, A. Refining Conversational Case Libraries. Proceedings of the 2nd International Conference on Case-Based Reasoning, ICCBR97, David B. Leake, Enric Plaza (Eds.). LNAI Vol. 1266, pp 265-278, Springer-Verlag, 1997.

[2] Ball, C. Screen-scraping with WWW::Mechanize www.perl.com/pub/a/2003/01/22/mechanize.html

[3] Bergmann, R., Richter, M. M., Schmitt, S., Stahl, A., Vollrath, I. (2001). Utility-Oriented Matching: A New Research Direction for Case-Based Reasoning. Proceedings of the 9th German Workshop on Case-Based Reasoning, GWCBR'01, Baden-Baden, Germany. In: H.-P. Schnurr, S. Staab, R. Studer, G. Stumme, Y. Sure (Hrsg.): Professionelles Wissensmanagement. Shaker Verlag. pp. 264-274.

[4] Bergmann, R. (2002). Experience Management: Foundations, Development Methodology, and Internet-Based Applications. Lecture Notes in Artificial Intelligence, Vol. 2432, Springer Verlag.

[5] Bradley K. & Smyth B. (2001). Improving Recommendation Diversity. Proceedings of the Twelfth Irish Conference on Artificial Intelligence and Cognitive Science, D. O'Connor (ed.) pp85-94, 2001.

[6] Branting, L. K. (2003). Learning Feature Weights from Customer Return-Set Selections. The Journal of Knowledge and Information Systems (KAIS). To appear, 2003.

[7] Burke, R., Hammond, K., & Young, B. (1997). The FindMe Approach to Assisted Browsing. IEEE Expert, 12(4), pages 32-40, 1997.

[8] Coyle, L., Cunningham, P. & Hayes, C. A Case-Based Personal Travel Assistant for Elaborating User Requirements and Assessing Offers. Proceedings of the 6th European Conference, ECCBR 2002, Susan Craw, Alun Preece (eds.). LNAI Vol. 2416 pp. 505-518, Springer-Verlag 2002.

[9] Coyle, L., Cunningham, P. & Hayes, C. (2002). Representing Cases for CBR in XML. Proceedings of 7th UKCBR Workshop, Peterhouse, Cambridge, UK.

[10] Foundation for Intelligent Physical Agents. Personal Travel Assistance Specification (Document No. XC00080). Carried forward from FIPA 1997 Specification 4 V1.0.
[11] Hayes, C., Cunningham, P., & Doyle, M. (1998). Distributed CBR using XML. In Proceedings of the KI-98 Workshop on Intelligent Systems and Electronic Commerce, number LSA-98-03E. University of Kaiserslauten Computer Science Department, 1998.
[12] Jarmulak, J., Craw, S. & Rowe, R. (2000). Genetic algorithms to optimize CBR retrieval. Proceedings of the 5th European Workshop on Case-Based Reasoning. LNAI Vol. 2416, pp 421-435, Springer-Verlag, 2000.
[13] McGinty, L. & Smyth, B. (2001). Collaborative CBR for Real-World Route Planning. In Proceedings of the 2001 International Conference on Artificial Intelligence (IC-AI'2001) Las Vegas, Nevada
[14] McGinty, L., & Smyth, B. (2002). Comparison-Based Recommendation. Proceedings of the 6th European Conference, ECCBR 2002, Susan Craw, Alun Preece (eds.). LNAI Vol. 2416, pp 575-589, Springer-Verlag, 2002.
[15] McSherry, D. (2002). Mixed Initiative Dialogue in Case-Based Reasoning. Proceedings of the Mixed-Initiative Case-Based Reasoning Workshop at ECCBR 2002.
[16] Minor, M., Wernicke, M. (2003). The Exchange of Retrieval Knowledge about Services between Agents. Wissensmanagement 2003: pp. 295-302
[17] Ricci, F., Arslan, B., Mirzadeh, N., Venturini, A. (2002). ITR: a Case-Based Travel Advisory System. Proceedings of the 6th European Conference, ECCBR 2002, Susan Craw, Alun Preece (eds.). LNAI Vol. 2416, pp 613-627, Springer-Verlag, 2002.
[18] Shimazu, H. (2001). ExpertClerk: Navigating Shoppers' Buying Process with the Combination of Asking and Proposing. Proceedings of the 17th International Joint Conference on Artificial Intelligence, IJCAI-01, Seattle, Washington, USA.
[19] Stahl, A. (2002). Defining similarity measures: Top-Down vs. bottom-up. Proceedings of the 6th European Conference, ECCBR 2002, Susan Craw, Alun Preece (eds.). LNAI Vol. 2416, pp 406-420, Springer-Verlag, 2002.
[20] Stahl, A. (2001). Learning Feature Weights from Case Order Feedback. Proceedings of the 4th International Conference on Case-Based Reasoning, ICCBR 2001
[21] Stahl, A., Gabel, T. (2003). Using Evolution Programs to Learn Local Similarity Measures. To appear in the proceedings of the 5th International Conference on Case-Based Reasoning, ICCBR 2003.
[22] Torrens, M., Faltings, B., & Pu, P. (2002). "Smart Clients: Constraint Satisfaction as a Paradigm for Scaleable Intelligent Information Systems". CONSTRAINTS 7(1), 2002, pp. 49-69
[23] Watson, I. & Gardingen, D. (1999). A Distributed Case-Based Reasoning Application for Engineering Sales Support. In, Proc. 16th Int. Joint Conf. on Artificial Intelligence (IJCAI-99), Vol. 1: pp. 600-605. Morgan Kaufmann Publishers Inc. ISBN 1-55860-613-0